Page:
Date: / /

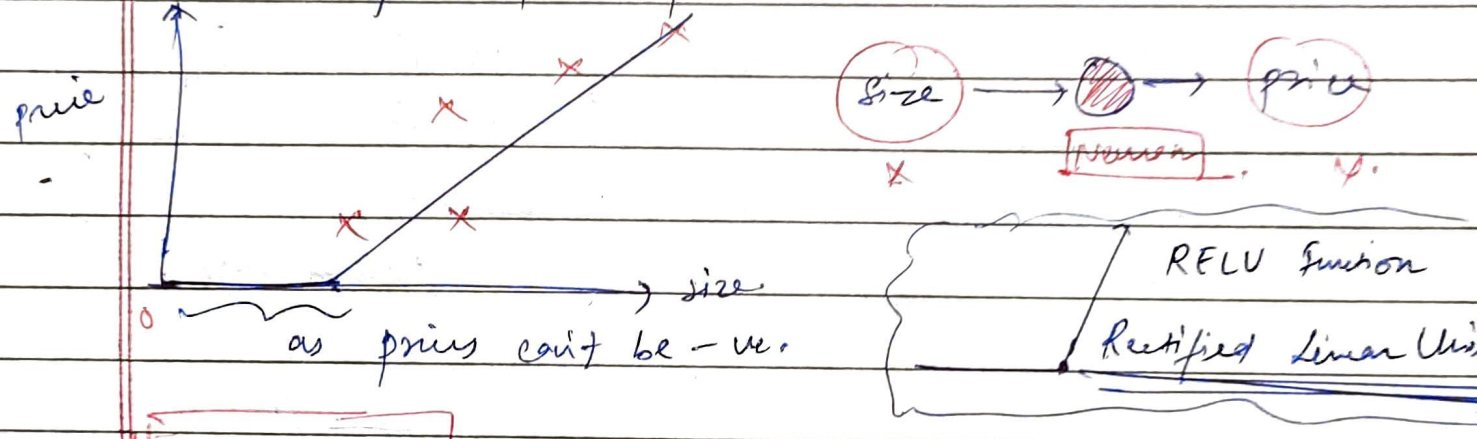
Course - 1 Neural network and deep learning

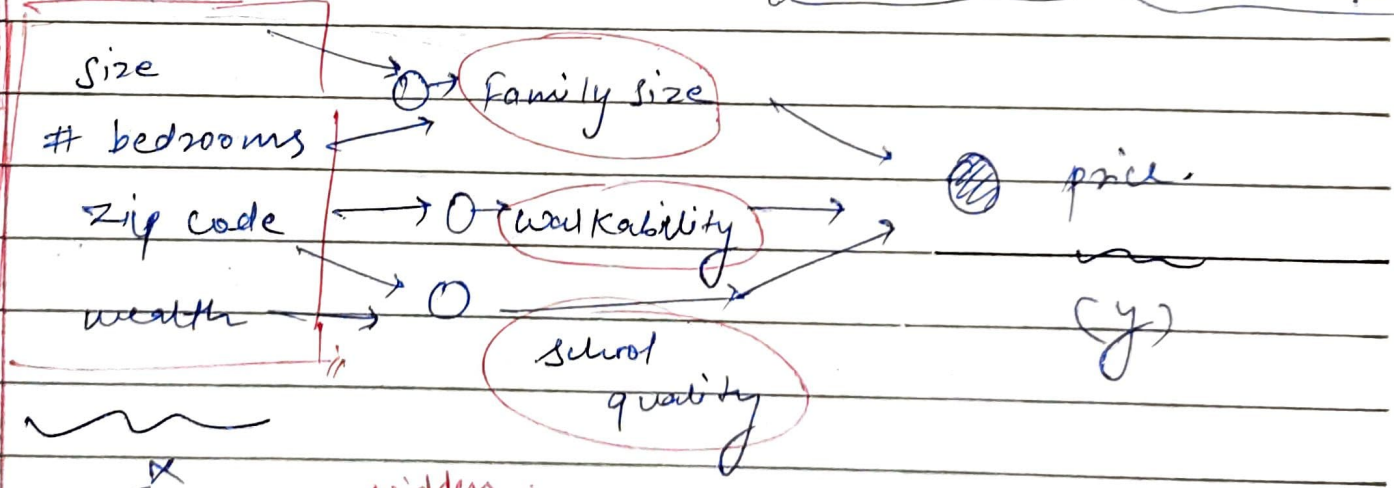
by Andrew ng

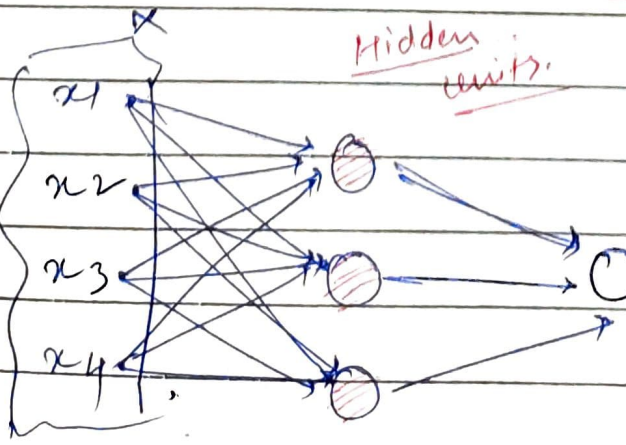
week - 1 INTRODUCTION TO DEEP LEARNING

## What is Neural n/w

→ deep learning refers to training neural n/w. ex in housing price prediction. :→



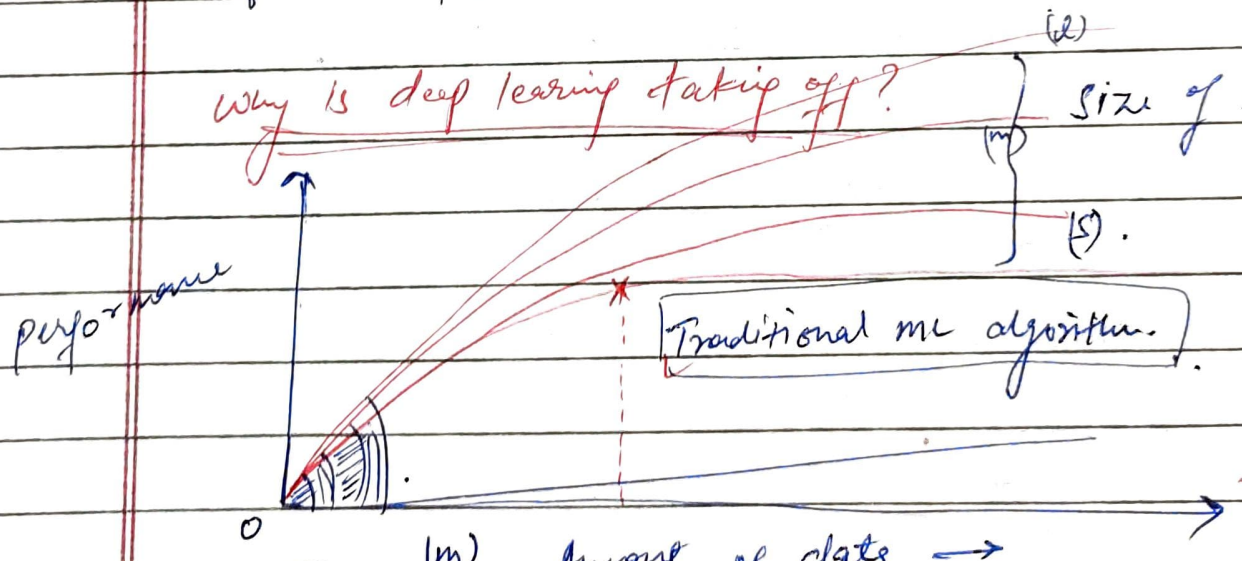
Page :
Date: / /

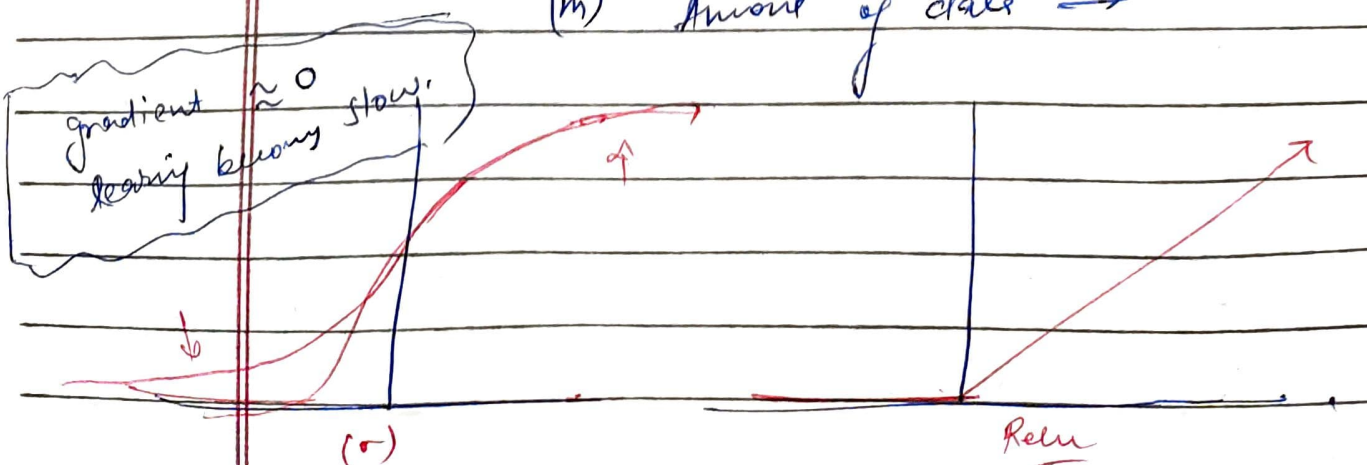
# Supervised Learning with neural networks.

## SUPERVISED LEARNING

| (X) | (Y) | Application. | |
|---|---|---|---|
| Home features | Price | Real estate | (Standard NN) |
| Ad, clicking. | 0/1 | online Adv. | (Standard NN) |
| Image | object (1..100) | photo tagging | CNN |
| Audio | Text transcript | Speech recog | RNN |
| English | Chinese. | Machine transl. | RNN |
| Image / Radar info | position. | Autonomous driving. | (Custom NN.) |

## Why is deep learning taking off?



# Deep Learning Specialization

1. Neural n/w and deep learning
2. Improving deep neural n/w.
3. Structuring your ML project.
4. Convolution neural network.
5. NLP: Building sequence model.

Page:
Date: / /

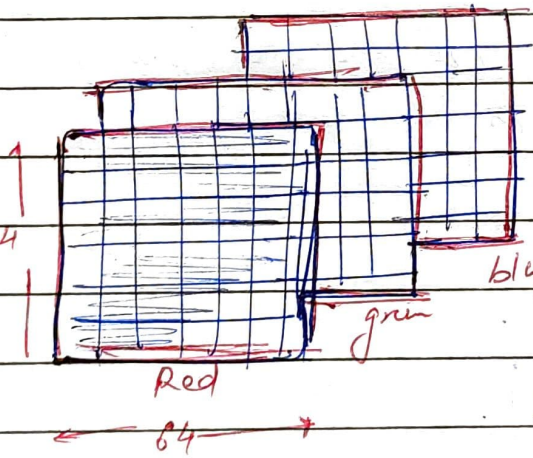
---

## Week-2 NEURAL N/W BASICS.

### Binary Classification


→ either x or !x
(1) (0).

### Storing an image.




To turn these pixel intensity values into feature vector, unroll all ~~feature~~ intensity values in feature vector.

$$x = \begin{bmatrix} 255 \\ 231 \\ \vdots \end{bmatrix}$$ feature vector.

* If image is 64×64

dimension of matrix = (64 × 64) × 3 = 12288.

$n / n_x = 12288.$

# Notation → $(x, y)$ $x \in \mathbb{R}^{n_x}$, $y \in \{0, 1\}$

→ m training example.

→ $m_{Train}$ → m test.

Page :
Date: / /

$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_m \\ | & | & & | \end{bmatrix} \uparrow n_x \downarrow$$

$\leftarrow$ m columns $\rightarrow$

$$X = \mathbb{R}^{n_x \times m}$$

$$Y = [\, y_1 \quad y_2 \quad \cdots \quad y_m \,]$$

$$Y \in \mathbb{R}^{1 \times m}$$

## Logistic Regression

Given $x$, algo: $\hat{y} = \text{prob.}(y=1 \mid x)$.

$x \in \mathbb{R}^{n_x}$

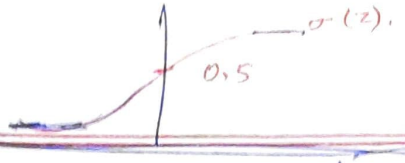
Parameters $= w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$.

Output : $\hat{y} = ?$

$$\hat{y} = w^T x + b.$$ (x).

$$0 \leq \hat{y} \leq 1$$

O/P = $\hat{y} = \sigma(w^T x + b)$




Page :
Date: / /

$\sigma(z) = \frac{1}{1+e^{-z}}$

if z if very large $e^{-z} \approx 0$

$\sigma(z) = \frac{1}{1+\approx 0} \approx 1$

if z is << small. then $\sigma(z) = 1/(1+\text{big})$

$\simeq 0$.

(4.18)

## Logistic Regression Cost function

$\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = 1/(1+e^{-z})$.

given $\{(x^1, y^1), (x^2, y^2) \ldots (x^m, y^m)\}$.

$z^{(i)} = w^T x^{(i)} + b$

### Loss / error function

$L(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$.

want $\log \hat{y}$ large

vice-versa

$y=1 \Rightarrow -\log \hat{y}$.

$y=0 \Rightarrow \log(1-\hat{y})$.

$\hat{y}$ = algorithm o/p

$y$ = original label.

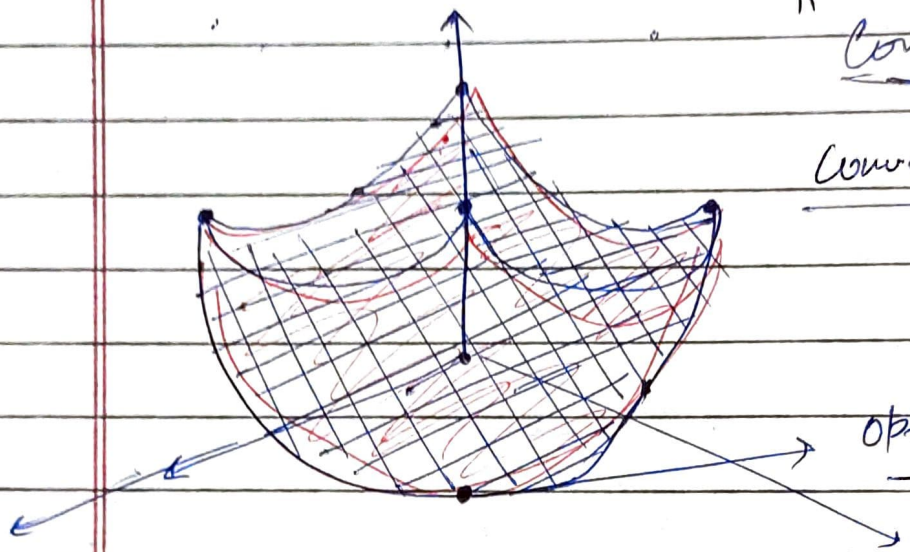
(*) Loss function was defined for a single training example.

(*) Cost function which measures over entire training set. $J(w,b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^i, y^i)$

$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i \log \hat{y}_{(i)} + (1-y_i)\log(1-y\,hat_{(i)})\right)$.

Page :
Date: / /

## Gradient Descent



Convex function

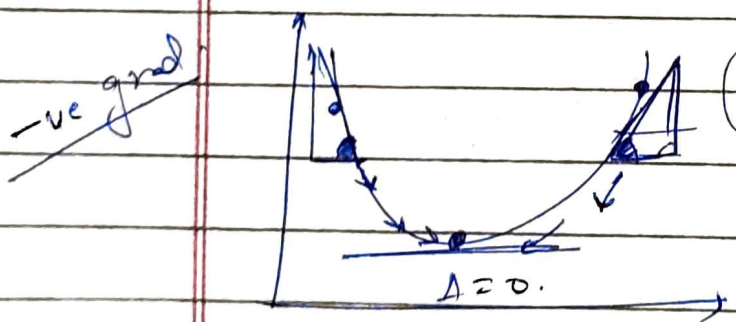
Converge to minima

optimum

Repeat until convergence :→

$$w := w - \alpha \frac{d J(w)}{dw}$$

learning rate.




$\frac{\partial J(w,b)}{\partial w}$

partial derivative

$$J(w,b) = \begin{cases} w := w - \alpha \dfrac{d J(w,b)}{dw} \\ b := b - \alpha \dfrac{d J(w,b)}{db} \end{cases}$$

dw

db

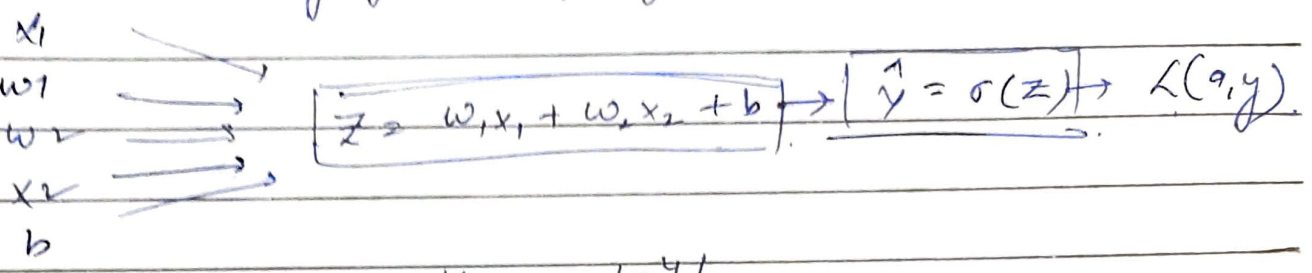
# Logistic Regression - Gradient descent

$z = w^T x + b$

$\hat{y} = a = \sigma(z)$

Page :
Date: / /

$\mathcal{L}(a, y) = -(y \log(a) + (1-y) \log(1-a))$.




"$da$" $= \frac{d}{da} \mathcal{L}(a,y) = -y/a + (1-y)/(1-a)$.

"$dz$" $= \frac{d\mathcal{L}(a,y)}{dz} = \frac{d\mathcal{L}}{da} \cdot \frac{da}{dz} = a - y$.

$\frac{d\mathcal{L}}{dw_1} = x_1 dz$, $dw_2 = x_2 dz$, $db = dz$.

## gradient descent with m examples

$J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(a_i, y)$.

$a_{(i)} = \hat{y}^{(i)} = \sigma(z_i) = \sigma(w^T x^{(i)} + b)$.

$dw_1\ (i)$
$dw_2\ (i)$
$db\ (i)$

$J = 0$ ; $dw = 0$ ; $db = 0$

for $i = 1$ ~~to~~ $m$ :→

$z^{(i)} = w^T x + b$

$a^{(i)} = \sigma(z^{(i)})$

$J\ += y^{(i)} \log(a^{(i)}) + (1-y^{(i)})(\log(1-a^{(i)}))$

$dz^{(i)} = a^{(i)} - y^{(i)}$.

$dw_1\ += x^{(i)} dz^{(i)}$.

$dw_2\ += x_2^{(i)} dz^{(i)}$.

$db\ += dz^{(i)}$,

$J /= m$

$n = 2$.

$dw_1 = \frac{dJ}{dw_1}$

$w_1 := w_1 - \alpha dw_1$

Page:
Date: / /

$w_2 := w_2 - \alpha dw_2$

$b := b - \alpha db$.

# Avoid explicit for loops and use Vectorization

## Vectorization

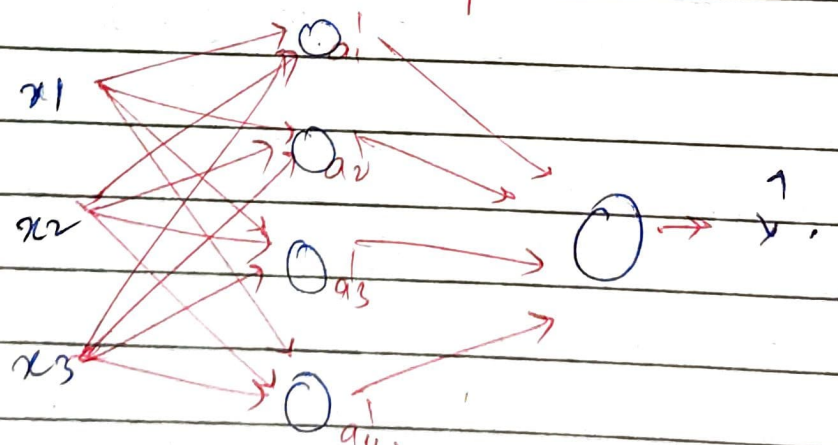
import numpy as np

$z = np.dot(w, x) + b$.

---

week-3 Shallow Neural Nw.

one hidden ~~training~~ layer



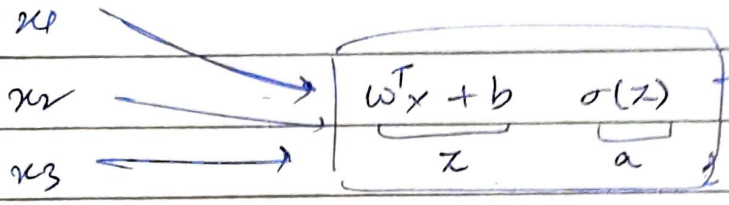

$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ \vdots \\ a_n^{[1]} \end{bmatrix}$

O ⟶ hidden layer ⟶ O ⟶ $\hat{y}$.

Input Layer

Page :
Date: / /

## Computing a ~~logistic regression~~ NN Output :-




$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

$a_i^{[2]}$ ⟶ layer

$a_i^{[2]}$ ⟶ node.

Vectorized form

$$\begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ \vdots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \end{bmatrix}$$

## Vectorization Across multiple examples.

$$z^{[1](1)} = w^1 x^1 + b^1, \quad z^{[1](2)} = w^1 x^2 + b^1,$$

$$z^{[1](3)} = w^1 x^3 + b^1.$$

$$w^{[1]} = \begin{bmatrix} — \\ — \\ — \end{bmatrix} \quad w^1 x^1 = \begin{bmatrix} \vdots \end{bmatrix} \quad w^1 x^2 = \begin{bmatrix} \vdots \end{bmatrix} \quad w^1 x^3 = \begin{bmatrix} \vdots \end{bmatrix}$$

$$w^{[1]} * \begin{bmatrix} | & | & | & \cdots \\ x^1 & x^2 & x^3 & \\ | & | & | & \end{bmatrix} = \begin{bmatrix} | & | & | \\ z^{[1]1} & z^{[1]2} & z^{[1]3} \\ | & | & | \end{bmatrix}$$

X

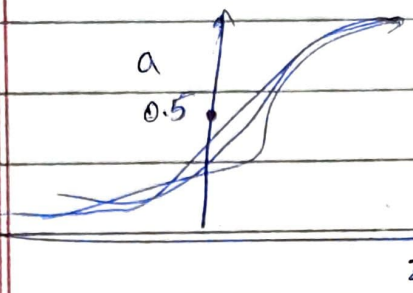
$X = A^{[0]}$

$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$.

Page :
Date: / /

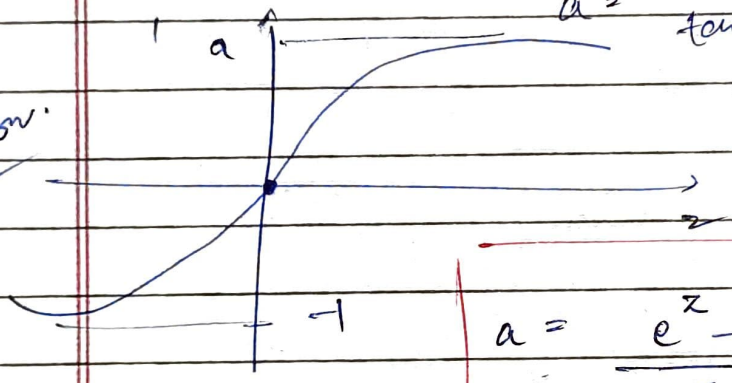
## Activation functions

Sigmoid function




$a = 1/1+e^{-z}$

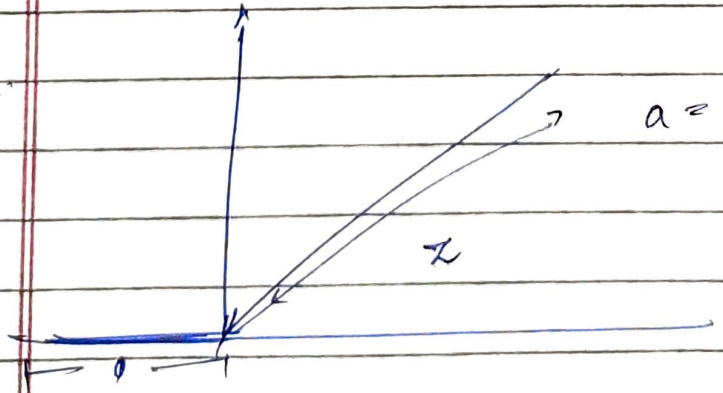
tanh function.




$$a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} = \tanh(z).$$

* tanh function almost everytime superior than Sigmoid function.

Relu function.

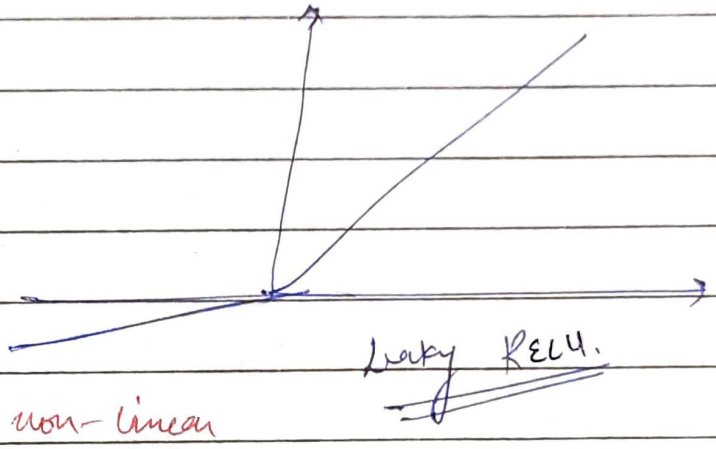


tanh : $a = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

Page :
Date: / /

$\tanh(z) = \frac{\sinh z}{\cosh z} = \frac{e^z - e^{-z}}{e^z + e^{-z}}$.

$= \frac{e^{2z} - 1}{e^{2z} + 1}$



Leaky ReLU.

Why do we need non-linear activation function ?

$a^{[1]} = z^{[1]} = w^{[1]} x + b^{[1]}$

$a^{[2]} = z^{[2]} = w^{[2]} a^{1} + b^{[2]}$

$= w^{[2]} \cdot (w^{(1)} \times b^{(1)}) + b^{[2]}$

$= (w^{[2]} \cdot w^{[1]}) x + (w^2 b^1 + b^2)$

$= w^1 x + b^1$

$g'(z)$ of $\tanh(z) = 1 - (\tanh(z))^2$

Relu function = $(0, z<0 ; 1, z>0)$

Sigmoid = $g(z)(1 - g(z))$.

gradient descent for neural n/w :-

Parameters = $w^1, b^1, w^2, x^2$.

Cost function = $J$ (Parameters)

Page :
Date: / /

$= \frac{1}{m} \sum_{i=1 \to n} \mathcal{L}(\hat{y}, y)$.

gradient Descent :-

repeat {

$dw^{[1]} = \frac{dJ}{dw^{[1]}}$

$db^{[1]} = \frac{dJ}{db^{[1]}}$

$w^{[1]} := w^{[1]} - \alpha\, dw^{[1]}$.

$b^{[1]} := b^{[1]} - \alpha\, db^{[1]}$.

$b^2$ ——— .

Backprop.

$dz^{[2]} = A^{[2]} - Y$

$dw^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$

$db^{[2]} = 1/m$ np.sum( $dz^{[2]}$, axis=1, inplace = True)

$dz^1 = w^{(2)T} dz^{(2)} * g^{(1)'}(z^{(1)})$.

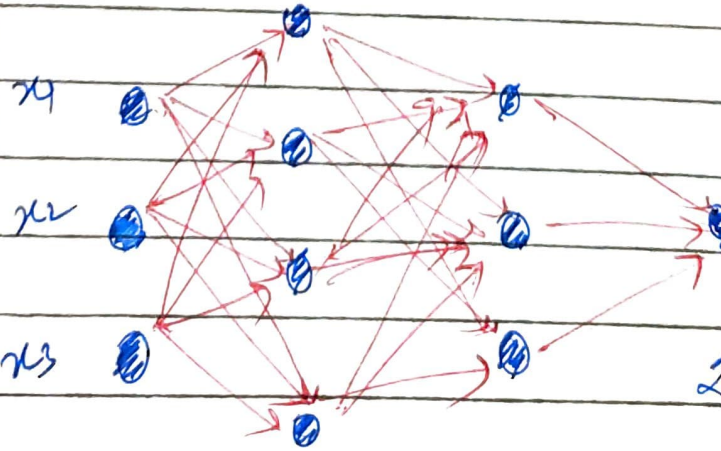
$dw^{(1)} = \frac{1}{m} dz^{(1)} x^T$.

$db^{(1)} = \frac{1}{m}$. np.sum($dz^1$, axis=1 keepdims = True ),

Week-4 — Deep Neural Networks.

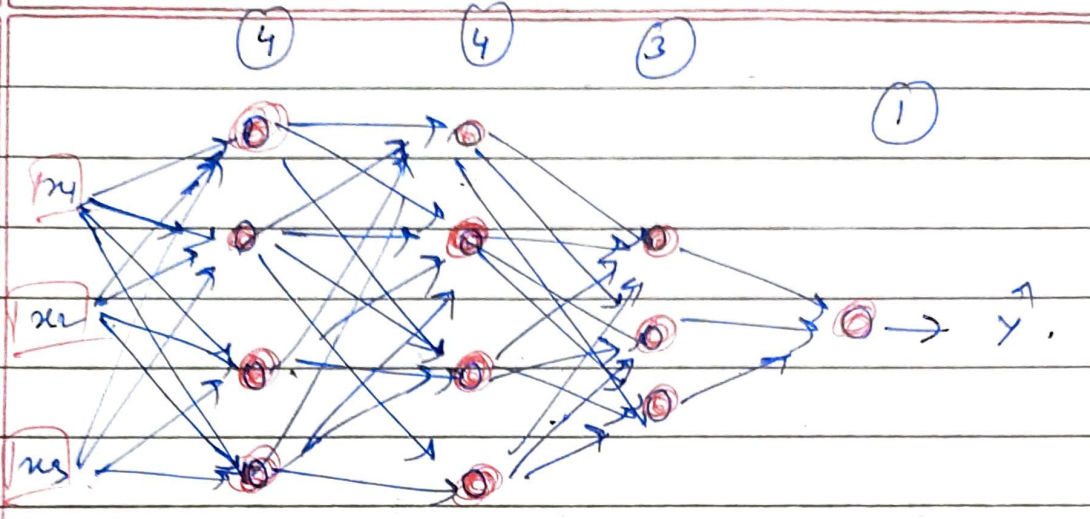
Lecture 1 — Deep L Layer neural network.

① Neural n/w with 2 hidden layers.




2 hidden layers

Page :
Date: / /




notations :)

$L$ :- No. of layers.

$n^{[l]}$ :- No. of units in layer $l$.

$a^{(l)}$ :- activation in layer $l$.

$a^{[l]} = g^{[l]}(z^{[l]})$

$w^{[l]}$ = weights for $z^{[l]}$.

$a^{[0]} = x$ , $a^{[L]} = \hat{y}$.

$n^{[1]} = 4$ | $n^{[4]} = 1$

$n^{[2]} = 4$ | $n^{[0]} = n_x = 3$.

$n^{[3]} = 3$

Week 2 Forward Propagation in Deep N/w

- for a single training example $x$.

$x$ :

$$z^{[1]} = w^{[1]} \cdot x + b^{[1]}$$
$$a^{[1]} = g^{[1]}(z^{[1]}).$$

] L1

$$z^{[2]} = w^{[2]} \cdot a^{[1]} + b^{[2]}$$
$$a^{[2]} = g^{[2]}(z^{[2]}).$$

] L2.

....

Page :
Date: / /

$$z^{(4)} = w^{(4)} \cdot a^{(3)} + b^{(4)}$$

$$a^{(4)} = g(z^{(4)})$$

$$= \hat{y}.$$

$x = a^{0}$. $\rightarrow$ $A^{0} = X$.

$$z^{[l]} = w^{[l]} \cdot A^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g(z^{[l]})$$

# Vectorized Version :-

for layer 1 ... n

can't have explicit for loop.

$$Z^{[1]} = W^{[1]} \cdot A^{0} + b^{[1]}$$

$$A^{[1]} = g^{[1]}(Z^{[1]})$$

$$Z^{[2]} = W^{[2]} \cdot A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(Z^{[2]})$$

$$\hat{y} = g(Z^{[4]}) = A^{[4]}$$

@ 6:35 → $a^{[L]} = g^{[L]}(z^{[L]})$.

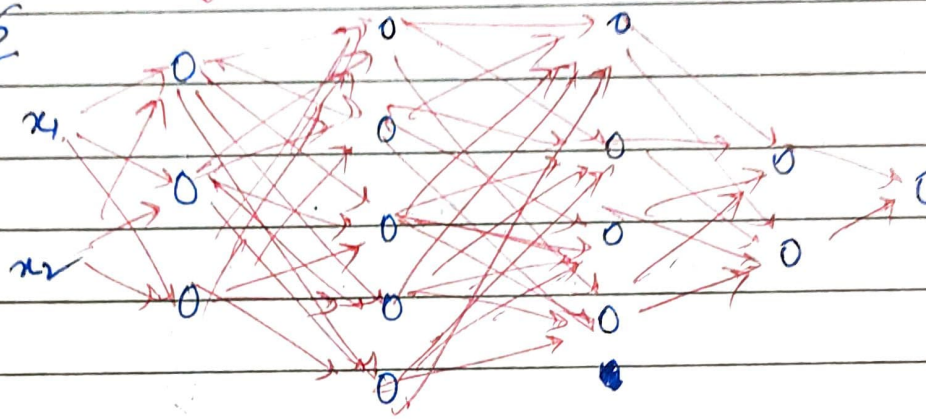
a, z → $(n^{[l]}, 1)$.

Page :

Date: / /

Lesson 3: Getting your matrix dimensions right

L = 5




$z^{[1]} = w^{[1]} \cdot x + b^{[1]}$.

(3,1) (3,2) (2,1)

$(n^{[1]}, 1) \cdot (n_1, n_0)(n_0, 1)$

| | |
|---|---|
| $n^{[1]} = 3$ | $n^{[3]} = 4$ |
| $n^{[2]} = 5$ | $n^{[4]} = 2$ |
| $n^{[5]} = 1$ | $n^0 = n_x = 2$ |

$z^{[1]} = (3,1) = (n^{[1]}, 1) = a^{[1]}$

→ $w^{[1]} = (3,2) = (n_1, n_0) \Rightarrow (n_l, n_{(l-1)})$.

$x = (2,1) = (n_0, 1)$

for dimension b :→

→ $b = (3,1) \Rightarrow (n^{[l]}, 1)$. and

→ $dw \simeq w$, $db \simeq b$

a and z have same dims.

Page :
Date: / /

$$Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$$

$(n^{[1]}, 1) \quad (n^{[1]}, n^{0}) \quad (n^{0}, 1) \quad (n^{[1]}, 1)$

$$Z = \begin{bmatrix} | & | & & | \\ z^{[1]} & z^{[1,2]} & \cdots & z^{[1,m]} \\ | & | & & | \end{bmatrix}_{(n^{[1]}, m)}$$

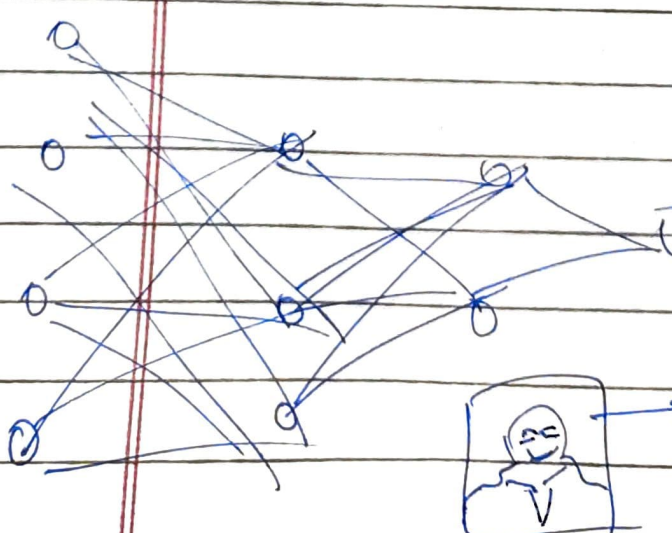
$Z^{[l]}, a^{[l]} : (n^{l}, 1)$

$Z^{[l]}, A^{[l]} : (n^{l}, m)$

if $l = 0 \quad A^{[0]} = X = (n_0, m)$

$dZ^{[l]}, dA^{[l]} : (n^{l}, m)$.

Lecture 4 — Why deep representations?






edge detection → High Level

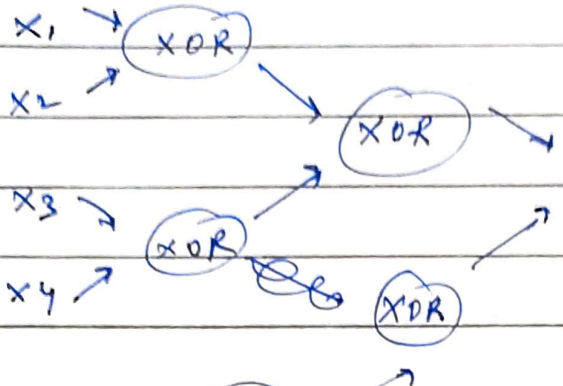
low level

Circuit theory and deep learning.

ex: $y = x_1 \text{ XOR } x_2 \text{ XOR } x_3 \text{ XOR } x_4 \ldots$

Page :
Date: / /

deep neural network

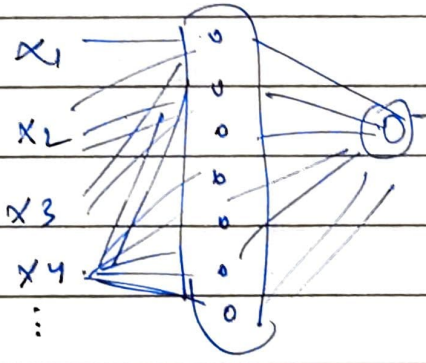



Complexity : $O(\log n)$.

~~true~~

$\simeq 2^{n-1}$.

Simple layer N/N.

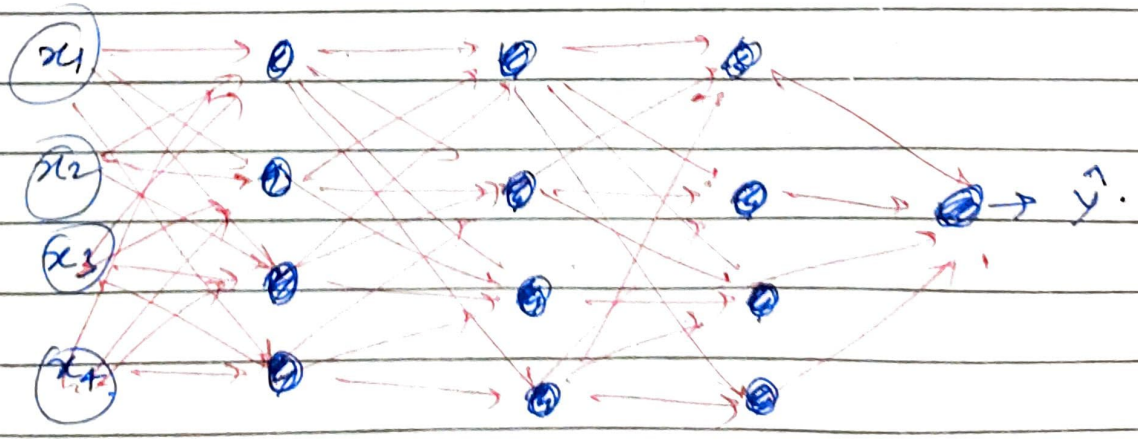



Complexity : $2^n$

Exponentially $\gg$ large

Lecture-5

## Building blocks of deep N/N



Page:
Date: / /

for Layer L : $w^{[L]}$ $b^{[L]}$

forward prop. : $a^{[l-1]}$, o/p : $a^{[l]}$

$z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b$, cache $z^{(l)}$.

$a^{[l]} = g(z^{(l)})$.

Backward prop. :→

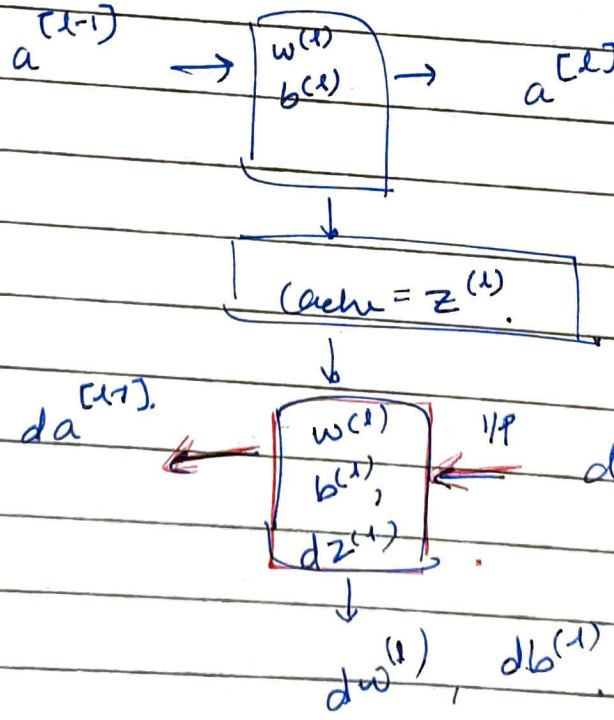
I/p : $da^{[l]}$ o/p : $da^{[l-1]}$.

cache $(z^{(l)})$. $dw^{(l)}$

$db^{(l)}$.

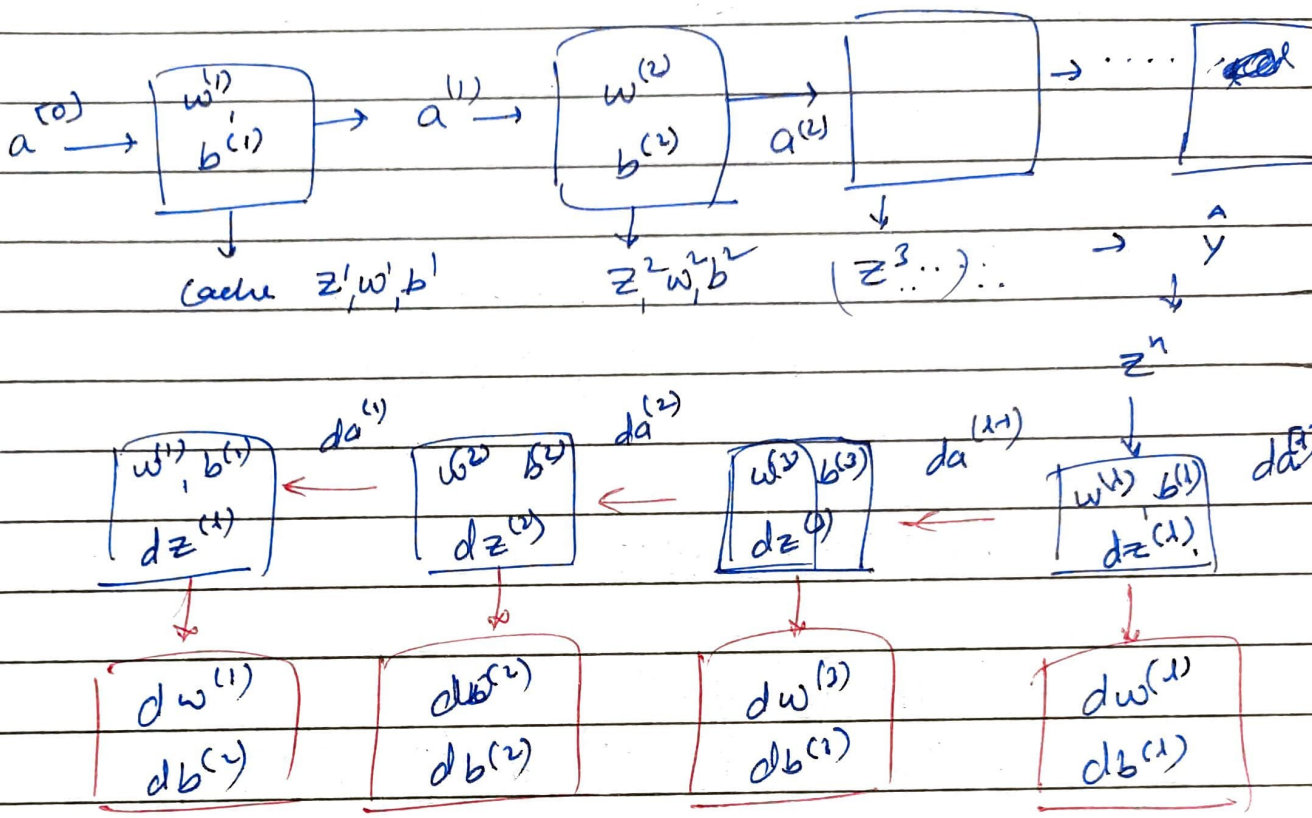
Summary

layer l.



Page :
Date: / /

forward prop.




backward prop.




$$w^{(l)} := w^{[l]} - \alpha . dw^{(l)}$$

$$b^{(l)} := b^{(l)} - \alpha . db^{(l)}$$

Lecture 6.

~~Parameters~~ & ~~Hyperparameters~~ ...

forward & back propogation

FORWARD PROP.

→ Input $a^{[l-1]}$

→ Output $a^{(l)}$, cache $z^{(l)}$

Page :
Date: / /

$$z^{(l)} = W^{(l)} \cdot a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g^{[l]}(z^{(l)})$$

Vectorized:

$$Z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$$

$$A^{[l]} = g^{[l]}(Z^{(l)})$$

## BACKWARD PROP.

→ I/P : $da^{[l]}$

O/P : $da^{[l-1]}, dw^{(l)}, db^{(l)}$

$$dz^{(l)} = da^{[l]} * g^{[l]'}(z^{(l)})$$

$$dw^{[l]} = dz^{(l)} \cdot a^{(l-1)}$$

$$db^{(l)} = dz^{(l)}$$

$$da^{[l-1]} = W^{T} \cdot dz^{[l]}$$

Vectorized

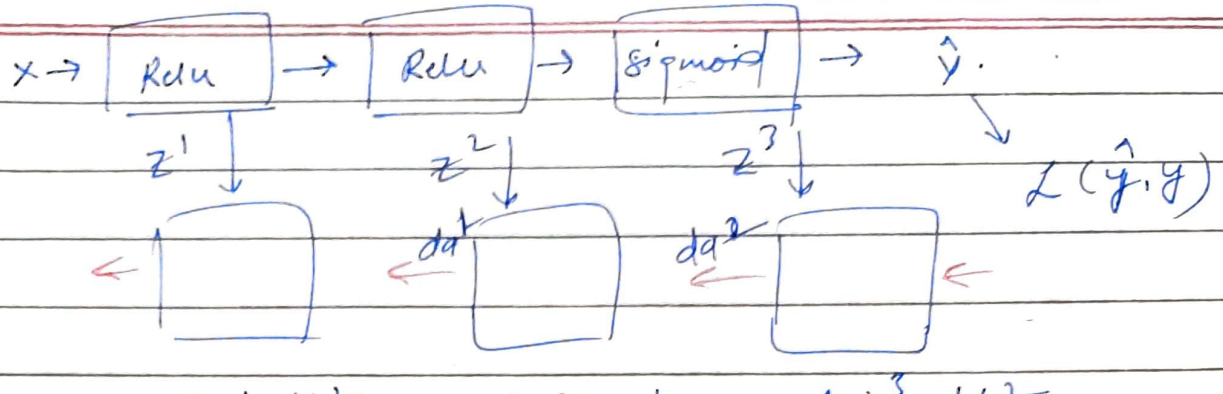
$$dz^{[l]} = dA^{[l]} * g^{[l]'}(Z^{(l)})$$

$$dw^{[l]} = \frac{1}{m} dz^{[l]} \cdot A^{[l-1].T}$$

$$db^{[l]} = 1/m \; np.sum(dz^{(l)}, axis=1, keepdims=1)$$

$$dA^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$$

Page :
Date: / /




$$da^{[L]} = -\frac{y}{a} + \frac{(1-y)}{(1-a)}$$

Lecture 7 — Parameters & Hyperparameters

PARAMETERS : $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}, \ldots, w^{[n]}, b^{[n]}$

HYPERPARAMETERS :- Learning rate ($\alpha$), #iterations, #hidden Layer / units, activation fx.

Eg:

$$dz^{[L]} = A^{[L]} - Y.$$

$$dw^{[L]} = \frac{1}{m} dz^{[L]} \cdot A^{[L-1]T}$$

$$db^{[L]} = \frac{1}{m} np.sum(dz^{(1)}, axis=1, keepdims=True)$$

$$dz^{[L-1]} = w^{(L)T} * g'^{[L]}(z^{(L-1)})$$